Problem 1

Basis functions can transform an input from a not linearly separable space to a linearly separable space. A neural network performs many such transformations, where the parameters of these basis functions are learned rather than hand-crafted. Using many basis functions, where each of them disentangles the dataspace even better a neural network can separate datapoints which could be not linearly separated in the original input space.

Problem 2

Problem 3

$$\begin{aligned}\frac{\partial}{\partial x}(\tanh(x)) &= \frac{\partial}{\partial x}(\frac{e^x - e^{-x}}{e^x + e^{-x}}) \\ &= \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \\ &= 1 - \tanh^2(x)\end{aligned}$$

It substantially simplifies the calculations during backpropagation when computing the local gradient at an intermediate node and when passing the gradient to the lower layers.

Problem 4

For deriving the new loss function under the specified requirements, we follow a similar approach as to the derivation in the case of using sigmoid (cf. Lecture Linear Classification, slide 28). We use $\tanh(\cdot)$ instead of $\sigma(\cdot)$. However to be able to still interpret the output as probabilities and define it as a Bernoulli experiment we need to rescale it to be in $[0, 1]$.

$$\begin{aligned}p(y = 1|x) &= \frac{\tanh(x) + 1}{2}, \quad y = 1 \\ p(y = -1|x) &= 1 - \frac{\tanh(x) + 1}{2}, \quad y = -1\end{aligned}$$

or more compactly (and useful later) as: $x^{\frac{1+y}{2}} \cdot (1 - x)^{\frac{1-y}{2}}$

$$\begin{aligned}E(W) &= -\sum_{i=1}^{N} \frac{1+y}{2}\log(\frac{\tanh(x_i, W) + 1}{2}) + \frac{1-y}{2}\log(1 - \frac{\tanh(x_i, W) + 1}{2}) \\ &= -\sum_{i=1}^{N} \frac{1+y}{2}\log(\frac{f(x_i, W) + 1}{2}) + \frac{1-y}{2}\log(1 - \frac{f(x_i, W) + 1}{2})\end{aligned}$$

Clearly the loss is zero for correctly classified datapoints and large for missclassified datapoints. Furthermore an output of $\tanh(x), x \to -1$ and $\tanh(x), x \to 1$ is transformed to a low (probability) value respectively high value. By construction the activation function to be chosen is $\tanh(\cdot)$

Problem 5

Problem 6

We would stop training approximately at iteration 50, because while the training error keeps decreasing we have an increase in the validation error, which is a sign of overfitting. After we take the weights of the network at this iteration we test it one final time on the test set. Note that we do not get the optimal test error as we would if we would continue with the training until iteration 80.

Problem 7

$$
\begin{aligned}
y &= \log(\sum_{i=1}^{N} e^{x_i}) \\
&= \log(\sum_{i=1}^{N} \frac{e^a}{e^a} e^{x_i}) \\
&= \log(e^a \sum_{i=1}^{N} e^{-a} e^{x_i}) \\
&= \log(e^a) + \log(\sum_{i=1}^{N} e^{-a} e^{x_i}) \\
&= a + \log(\sum_{i=1}^{N} e^{x_i - a})
\end{aligned}
$$

Problem 8

$$
\begin{aligned}
\frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} &= \frac{e^a}{e^a} \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} \\
&= \frac{e^{x_i - a}}{\sum_{i=1}^{N} e^{x_i - a}}
\end{aligned}
$$

Problem 9

The binary cross-entropy is equivalent to

$$
\begin{aligned}
-[y\log(\sigma(x)) + (1-y)\log(1-\sigma(x))] &= -[y\log(\frac{e^x}{1+e^x}) + (1-y)\log(1-\frac{e^x}{1+e^x})] \\
&= -[y\log e^x - y\log(1+e^x) + (1-y)(-\log(1+e^x))] \\
&= -[yx - \log(1+e^x)] \\
&= -yx + \log(1+e^x) \\
&= -yx + \log(e^x(e^{-x}+1)) \\
&= x - yx + \log(1+e^{-x}) \\
&= \max(x,0) - yx + \log(1+e^{-|x|}) \qquad \text{(see below)}
\end{aligned}
$$

The binary cross-entropy and in general any well-defined loss function should satisfy the following properties : For correctly classified datapoints we want to have zero loss and for incorrectly classified points, we want to have a large loss. This can be easily verified for the binary cross-entropy, e.g. if x is a large positive score and y = 0, i.e. x is missclassified then we get a large loss value, contrary if y = 1 we get zero loss, as expected. Our stable binary cross-entropy should also satisfy this properties.
For $x \to -\infty$ we get an overflow when implementing it on computers. To overcome this issue, we can map any negative x to a positive value using the abs$(\cdot)$ function. To ensure that for missclassified datapoints we penalize with a non-zero loss we have to add the term $\max(x, 0)$ to the loss function (Note that in the casse of correctly missclassified datapoints this term either is zero or cancels out with yx, which ensures zero loss).